A Simple Method for Citation Metadata Extraction using Hidden Markov Models

Erik Hetzner
California Digital Library
415 20th St
Oakland, CA 94720
erik.hetzner@ucop.edu

ABSTRACT

This paper describes a simple method for extracting metadata fields from citations using hidden Markov models. The method is easy to implement and can achieve levels of precision and recall for heterogeneous citations comparable to or greater than other HMM-based methods. The method consists largely of string manipulation and otherwise depends only on an implementation of the Viterbi algorithm, which is widely available, and so can be implemented by diverse digital library systems.

Categories and Subject Descriptors

H.3.7 [**Information storage and retrieval**]: Digital Libraries—*systems issues*

General Terms

Algorithms

Keywords

Citation Management, Metadata Extraction

1. INTRODUCTION

Digital libraries are often confronted with the problem of turning a textual citation into a more structured reference. A structured reference can be used to aid the tasks of citation grouping, and also mapping of the citation graph, which can provide a basis for data-mining of research papers. Although it is trivial for a human to divide a citation into its constituent fields, it is not an easy task to make a computer do the same. The grammars used to produce citations are complex, varied, and not known in advance. Various rule- and knowledge-based [13, 7, 9], and machine-learning techniques [2, 12, 18, 22, 6, 4, 11] have been developed to extract metadata from citations. This paper describes a method for extracting structured references from plain text citations which is relatively simple and achieves acceptable precision and recall. Additionally, we feel that this method has potential for improvement. The engine of the method is a hidden Markov model which is learned by training on a small (less than 500) set of citations which have been marked up by hand.

2. CITATIONS AND REFERENCES

We use *citation* to mean the free-form string, referencing a particular bibliographic item, which a human can use to locate that item. A *reference* is the more abstract structure which can be used, in combination with a grammar, to generate a citation. Our model of a reference, which we attempt to recover from a given citation, is of a set of labeled fields, each of which is repeatable. Each author, for instance, is represented by a field labeled "author" which contains a string used to identify that author. This model is somewhat more strict than the BibTEX-influenced models which do not allow repeated fields and instead combine all authors into a single labeled field.

3. HIDDEN MARKOV MODELS

A discrete-emission hidden Markov model (HMM) [5] is a probabilistic model in which transitions between a finite set of states are accompanied by the emission of a symbol from a finite alphabet. They are described as "hidden" because the emission sequence is known, and from this sequence the task is to determine the state sequence which generated it.

An HMM begins in some starting state, q_S, and then transitions to a new state q_2, with probability $P(q_2|q_S)$, in the process emitting a symbol σ_S with a probability given by $P(\sigma_S|q_S)$. It continues in this way until it reaches an end state q_E, having emitted a sequence of symbols X.

The Viterbi algorithm [5] makes use of the Markov property of an HMM (that the next state transition and symbol emission depend only upon the current state) to determine, in linear time with respect to the length of the emission sequence, the most likely path through the states of a model which might have generated a given sequence. The Viterbi algorithm is widely known and used, and implementations of the Viterbi algorithm are available for many systems, including free implementations for Perl [10], Java [17], C/C++/Python [1].

HMMs have been used in the past for digital signal procession, speech recognition, bioinformatics and information extraction [5, 19, 14, 3, 21]. Although HMMs have been used before for metadata extraction from citations [22, 6, 18, 4, 11], our method achieves comparable or greater accuracy and does not depend on multiple HMMs, as in [6], a more complex bigram HMM [22], or a modified Viterbi algorithm [22]. It bears many similarities to [11], although developed independently; however, it does not make use HTML tags, and uses a different model, along with a reduced alphabet. It seems to achieve greater or comparable precision and accuracy to other HMM-based methods, although only [18] records measurements against the data set that we used.

4. DEFINING THE MODEL

A discrete-emission, first-order hidden Markov model lies at the heart of this method. An HMM is defined by the set of states, the alphabet of symbols, the state transition probability matrix, the emission probability matrix, and the initial state probability matrix [5]. In our case, the set of states Q is derived from the labels of fields that we wish to extract from the citations. The alphabet of symbols Σ is hand-built from individual words, word classes, punctuation, and word features. The probabilities are derived from training data.

4.1 The states

To build our set of states Q we begin with the labels of the fields which we wish to extract from each citation. What label a field has depends upon the underlying reference model. For instance, conference proceedings might be considered a separate label from a book title, or they might be considered the same. Some labels are used in certain sets of citations but not in others. In our case we used the following labels, present in the Cora dataset, with the addition of (issue) number (which in the Cora dataset was subsumed under volume): author, booktitle, date, editor, institution, journaltitle, location, note, number, pages, publisher, techtitle, title, and volume. In [20] it is noted that having one state per label is not optimal, and we have found this to be the case. We define two states in Q for each label: a "first" state and a "rest" state. So in the case of the "author" label, we have an a_f state and an a_r state. Other numbers of states were tried, but having a first and rest gave the best results.

We further defined a set of "separator states", which represent words and punctuation that are not part of fields, but which act to differentiate fields. For instance, authors' names are often separated by a comma "," or the word "and", and the set of authors is often terminated by a period ".", which also indicates the beginning of the title of a paper. The marks are not part of the labeled fields which we wish to extract, but are generated by the citation grammar and are intended to aid the reader in differentiating the parts of a citation. In order to exploit them, we define a set of separator states between each pair of fields. For example, the state which separates an author's name and the title state is the $a|t$ state. These separators states are non-active and do not contribute to labeled fields, although they are used to split repeated, adjoining fields with the same label. Finally, we define a non-active terminating "end" state.

4.2 The alphabet

Selection of a useful alphabet, Σ, is essential to acceptable performance of a hidden Markov model. If we take as our alphabet all words in the English language we end up with a dictionary which contains many words which our training documents will not contain, and we will be required to use complicated smoothing mechanisms to obtain useful probability matrices. Additionally, if we use the set of English words we will not exploit the relations between certain words, such as the months of the year, to help our model out. Instead, we have found it best to define a mapping of tokens to a smaller alphabet of symbols. This alphabet is composed of symbols which represent punctuation, particular words, classes of words, and word features.

We first define a set of symbols to represent various punctuation, which in our method are considered to be individual tokens. The tokens that we map from include the comma ",", period ".", hyphen "-", and so on. Each of these characters maps to one symbol. Unusual punctuation is mapped to a single miscellaneous punctuation symbol.

Next we define a set of symbols to represent certain key words. For instance, it is clear that the word "proceedings" and the abbreviation "proc" are important key words which can help identify conference proceedings. So we define a single symbol to represent this particular word/abbreviation. Other tokens mapped to a single symbol include "press", "university", and "report".

We further define a small number of word classes representing related words. For instance, it is clear that there exists a relationship between the months of the year, so a single symbol is defined to represent the months.

Finally, we define a set of symbols to match words features for the words which have not yet been matched. These features include an uppercase word, a lowercase word, a titlecase word, a string of numerals of length 4, and so forth.

4.3 Building the parameters

To build our model's probability matrices we use fully labeled training data. In this data each state sequence and emission sequence is completely defined. To calculate the transition matrices we use the method described in [20]. For the probability of each transition from state q_n to q_m, we calculate:

$$P(q_m|q_n) = \frac{c(q_n \rightarrow q_m)}{\sum_{q \in Q} c(q_n \rightarrow q)}$$

where $c(q \rightarrow q')$ is the count of occurrences in the training date of a transition from state q to state q'.

We use the same method to define the emission probabilities, $P(\sigma_n|q_n)$, for each $q_n \in Q$ and $\sigma_n \in \Sigma$, and the start probability, $P(q_S = q_n)$ for each $q_n \in Q$.

4.4 Naive smoothing

Because some state transitions and some symbol emissions will not be encountered in the training data, we will need to smooth our data somewhat. We use a very naive method, setting to a low constant (10^{-7}) the probabilities of events which previously had zero probability. In order to ensure that the sums of $P(q'|q)$ and $P(\sigma|q)$ are 1 for all values of q' and σ we subtract from the probability of all originally non-zero probability events the value $\frac{n}{m}10^{-7}$ where n is the count of originally zero-probability events and m the count of originally non-zero-probability events. If we have a state for which we have no training data for transitions or emissions, we use the uniform distribution.

5. APPLYING THE METHOD

Having built our model, we now describe the method by which a string of characters representing a citation is turned into a structured reference.

Table 1: Precision and recall

	Token			Field			Token (stripped)			Field (stripped)		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
author	98.0	98.9	98.4	88.0	90.6	89.3	99.5	99.6	99.5	94.0	96.8	95.4
booktitle	89.4	91.4	90.4	62.5	67.2	64.7	90.4	93.8	92.1	70.8	76.1	73.4
date	94.2	97.3	95.7	89.9	94.3	92.0	96.7	99.4	98.0	94.6	99.3	96.9
editor	86.3	90.3	88.3	54.8	63.9	59.0	89.2	92.2	90.7	54.8	63.9	59.0
institution	61.2	88.2	72.3	44.4	57.1	50.0	61.9	92.9	74.3	66.7	85.7	75.0
journaltitle	88.8	70.6	78.7	88.4	74.5	80.9	89.0	73.0	80.2	88.4	74.5	80.9
location	76.5	69.0	72.6	56.8	56.8	56.8	79.2	76.0	77.6	73.0	73.0	73.0
note	29.5	66.7	40.9	33.3	50.0	40.0	34.1	75.0	46.9	33.3	50.0	40.0
number	95.8	100.0	97.9	95.5	100.0	97.7	95.7	100.0	97.8	95.5	100.0	97.7
pages	99.2	98.0	98.6	95.3	96.5	95.9	98.8	98.2	98.5	95.3	96.5	95.9
publisher	72.8	83.8	77.9	68.4	78.8	73.2	73.2	83.3	77.9	68.4	78.8	73.2
techtitle	93.2	95.3	94.3	75.0	75.0	75.0	93.1	93.1	93.1	75.0	75.0	75.0
title	98.7	95.5	97.1	88.5	87.2	87.9	99.2	96.6	97.9	95.0	93.6	94.3
volume	92.5	75.4	83.1	91.8	76.3	83.3	96.1	81.7	88.3	95.9	79.7	87.0
all	93.4	93.3	93.3	83.3	84.9	84.1	94.3	94.4	94.4	88.3	90.0	89.2
macro average	84.0	87.2	84.7	73.8	76.3	74.7	85.4	89.6	86.6	78.6	81.6	79.8
whole instance		45.8			45.8			65.5			61.3	

5.1 Tokenizing a citation

The first step of transforming the given string of characters that make up a citation into a string of symbols is to tokenize the citation. We consider as a separate token any character in the Unicode punctuation category [8], with the addition of few additional characters, including "+", "|", and "˜", which are not considering punctuation in the Unicode standard. We further consider as a token any string of characters not previously matched, separated from its neighbors by whitespace. At the end of this step we have a token sequence, $\langle w_1, \ldots, w_n\rangle$.

5.2 Processing tokens into symbols

For each of our emission symbols we define one or more regular expressions which will can be used to map a token to a symbol. For instance, the regular expression `^[Jj]an (uary)?` is one regular expression defined for the month symbol. We apply, in order of precedence, this sequence of regular expressions to each token from the token sequence previously obtained, giving us the symbol sequence: $\langle x_1, \ldots, x_n\rangle$.

Because our mapping from tokens to symbols is one way, we must retain our original token sequence in order to later zip together the states with the original token.

5.3 Viterbi

At this stage we can pass our sequence of symbols $\langle x_1, \ldots, x_n\rangle$ and our model, as built above, to our implementation of the Viterbi algorithm. We use the Python bindings of the GHMM [1] library for this step. The result is the sequence of states, $\langle q_1, \ldots, q_n\rangle$, most likely to have produced the symbol sequence which represents our citation.

5.4 Joining the citation fields

Now we can zip the list of states obtained above with our original token sequence, obtaining a sequence of state-token tuples, $\langle (q_1, w_1), \ldots, (q_n, w_n)\rangle$. These states can be trivially joined into labeled fields by joining each token in the list with the previous one, if the states represent the same label; that is, are either a first or rest state for that label. These tokens can be joined together with whitespace to obtain each labeled field string.

6. SOME RESULTS

We tested our method against the Cora dataset of labeled citations [16, 15], after reworking the data somewhat to fit our model. These changes include: breaking the author and editor fields into separate fields for each person; moving some separator tokens between labeled states; correcting some errors in labeling; and converting some text which was formerly marked as a note into labeled fields. We made these changes to every entry in the set, and trained from the first 350 entries, testing against the final 142 entries.

We measured the per-field and per-token precision and recall for each label. The results of this measurement are in Table 1. The per-token measurements test the precision and recall of each token marked with a label state. The per-field measurements test the precision and recall for each joined and labeled field.

Because accurate recovery of punctuation present in the original citation may or may not be useful for a given task, we also measured the per-token and per-field performance after stripping out punctuation. For the per-token measure, this meant ignoring all punctuation tokens when measuring performance. For the per-field measure, after grouping the results into labeled fields, we dropped any leading or trailing punctuation before considering if fields were a match. Since extra or missing punctuation at the beginning or end of a field is a common error in our method, this improves the performance with what may be a tolerable loss in accuracy.

At the bottom are the macro average, the total for all fields, and the special "whole instance" measure. Whole instance measures the percentage of references whose machine-labeled tokens or fields completely match the hand-labeled reference. With non-punctuation stripped data the whole instance accuracy is the same per-token and per-field. With the stripped data it is higher in the per-token comparison, because in this comparison we ignore some non-equal punctuation which might be used to split fields.

It is worth noting that Cora dataset is a very heterogeneous set of citation data, and that preliminary results with this method tested against the more homogeneous health sciences dataset used in [7] give whole instance accuracy greater than 90%.

While these results are inferior to those obtained by the use of conditional random fields in [18], we feel that there may still be a place for HMM-based methods for citation parsing, due to their relative ease of use, availability of implementations, and possibility for improvement.

7. IMPROVEMENTS

It is certain that the symbol alphabet and the corresponding regular expressions used to map from tokens to symbols are suboptimal. Techniques could be used to optimize this mapping. We suspect that the use of domain knowledge in mapping between tokens and symbols would aid in accuracy. For example, dividing words into common words and proper names might improve labeling of authors and titles.

Our mapping from field labels to model states is likewise crude. It would be useful to exploit some of the methods described in [20] to build better models from the training data.

The employment of a second-order HMM, and a suitably extended Viterbi algorithm would likely also be an improvement on the accuracy of this method.

8. CONCLUSION

We have shown that it is possible to achieve good results in citation metadata extraction using hidden Markov models, through a careful selection of states and symbols, reducing the size of the symbol alphabet, using two states for each label, and making use of separator states. The simple nature of this method and the easy availability of Viterbi algorithm implementations should allow digital library implementers to employ this method in their systems.

Thanks to the anonymous reviewers for their valuable comments. The code used to generate the results in this paper may be found at:
`http://purl.net/net/egh/hmm-citation-parser/`

9. REFERENCES

[1] Algorithmics group. Max Planck Institute for Molecular Genetics. GHMM: A LGPL-licensed hidden markov model library. `http://ghmm.sourceforge.net/`, 2008.

[2] D. Besagni and A. Belaïd. Citation recognition for scientific publications in digital libraries. In *Proc. of the First Intl. Workshop on Document Image Analysis for Libraries*, pages 244–252. IEEE Computer Society, 2004.

[3] D. Bikel, S. Miller, R. Schwartz, and R. Weischedel. Nymble: a high-performance learning name-finder. In *Proc. of the 5th Conf. on Applied Natural Language Processing*, pages 194–201, Washington, D.C., 1997.

[4] V. R. Borkar, K. Deshmukh, and S. Sarawagi. Automatic segmentation of text into structured records. In *Proc. of the 2001 ACM SIGMOD Intl. Conf. on Management of Data*, pages 175–186, 2001.

[5] E. Charniak. *Statistical language learning*. MIT Press, Cambridge, Mass., 1999.

[6] J. Connan and C. Omlin. Bibliography extraction with hidden markov models. Technical Report US-CS-TR-00-6, Department of Computer Science, University of Stellenbosch, Feb. 2000.

[7] E. Cortez, A. S. da Silva, M. A. Gonçalves, F. Mesquita, and E. S. de Moura. FLUX-CiM: flexible unsupervised extraction of citation metadata. In *Proc. of the 7th ACM/IEEE Joint Conf. on Digital Libraries*, pages 215–224, Vancouver, BC, Canada, 2007. ACM.

[8] M. Davis and K. Whistler. Unicode character database. `http://www.unicode.org/Public/UNIDATA/UCD.html`, 2008.

[9] M.-Y. Day, R. T.-H. Tsai, C.-L. Sung, C.-C. Hsieh, C.-W. Lee, S.-H. Wu, K.-P. Wu, C.-S. Ong, and W.-L. Hsu. Reference metadata extraction using a hierarchical knowledge representation framework. *Decision Support Systems*, 43:152–167, Feb. 2007.

[10] K. Dejonghe. Algorithm::Viterbi. `http://search.cpan.org/~koen/Algorithm-Viterbi-0.01/lib/Algorithm/Viterbi.pm`, Nov. 2006.

[11] J. Geng and J. Yang. AUTOBIB: Automatic extraction of bibliographic information on the web. In *Proc. of the Intl. Database Engineering and Applications Symposium*, pages 193–204. IEEE Computer Society, 2004.

[12] M. Krämer, H. Kaprykowsky, D. Keysers, and T. Breuel. Bibliographic meta-data extraction using probabilistic finite state transducers. In *Proc. of the Ninth Intl. Conf. on Document Analysis and Recognition*, pages 609–613. IEEE Computer Society, 2007.

[13] S. Lawrence, C. L. Giles, and K. Bollacker. Digital libraries and autonomous citation indexing. *IEEE Computer*, 32:67–71, 1999.

[14] T. R. Leek. *Information extraction using hidden Markov models*. Masters, University of California, San Diego, 1997.

[15] A. McCallum. Andrew McCallum's code and data. `http://www.cs.umass.edu/~mccallum/code-data.html`, 2005.

[16] A. McCallum, K. Nigam, J. Rennie, and K. Seymore. A machine learning approach to building domain-specific search engines. In *The 16th Intl. Joint Conf. on Artificial Intelligence*, 1999.

[17] R. A. Milowski. jHMM. `http://www.milowski.com/software/jhmm/`, 2005.

[18] F. Peng and A. McCallum. Accurate information extraction from research papers using conditional random fields. In *Proc. of Human Language Technology Conf. and North American Chapter of the Association for Computational Linguistics*, 2004.

[19] M. Perrow and D. Barber. Tagging of name records for genealogical data browsing. In *Proc. of the 6th ACM/IEEE-CS Joint Conf. on Digital Libraries*, pages 316–325, Chapel Hill, NC, USA, 2006. ACM.

[20] K. Seymore, A. McCallum, and R. Rosenfeld. Learning hidden markov model structure for information extraction. In *Workshop on Machine Learning for Information Extraction*, 1999.

[21] A. Viterbi. A personal history of the viterbi algorithm. *IEEE Signal Processing Magazine*, 23:120–142, 2006.

[22] P. Yin, M. Zhang, Z. Deng, and D. Yang. Metadata extraction from bibliographies using bigram HMM. In *Proc. of the 7th Intl. Conf. on Asian Digital Libraries*, LCNS 3334, pages 310–319, 2004.